

Developing for IBM Digital Experience with Web Developer Skills and Tools Using IBM Script Portlet

With Updates for Script Portlet 1.3



Topics

★ *new* = new for Script Portlet 1.3

- Script-based applications and the Script Portlet approach
- Working with the Script Portlet editor
 - Inserting WCM tags and storing in shared libraries ★ *new*
- Building applications with external editors and tools
 - Using “sp push” to create Script Portlet applications ★ *new*
- Complex applications, frameworks, and the “single page application” model
- Making applications available on the site toolbar ★ *new*
- Using command line features and working with source code systems ★ *new*
- Quick steps for getting started and downloadable samples ★ *new samples*
- Customizable applications and accessing data and services

Why script-based applications

- Web programming skills – using HTML/JS/CSS – are very prevalent and widely available
- Many JavaScript libraries and frameworks are available and widely known (jQuery, Backbone.js, AngularJS, Knockout, ExtJS, e.g.)
- A client-side application architecture – running JS on the client, accessing data via REST services – can have performance and user experience benefits



BACKBONE.JS



ANGULARJS
by Google

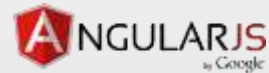
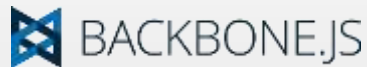


The Script Portlet approach

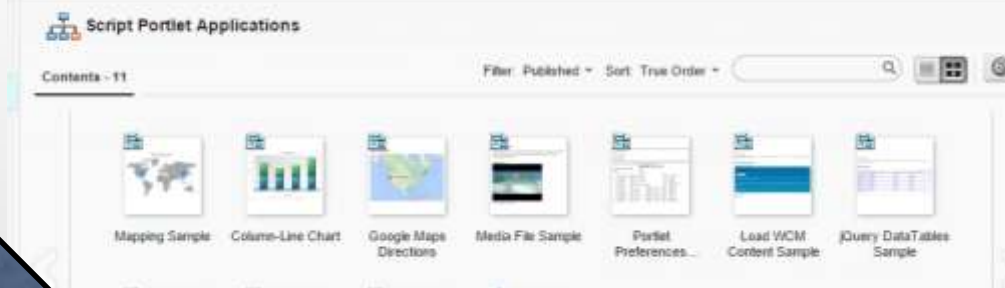
- Applications are built with standard web technology: HTML, JS, CSS, etc.
 - No Java or JEE skills needed
- Application artifacts are stored and managed in Web Content Manager
 - No code deployment required!
 - WCM features such as projects and workflow can be used to manage applications
- You can build applications from the browser with the Script Portlet Editor, or you can use your favorite external editor
- Portal's key value-add features are available when you want them
 - Portlet preferences, public render parameters, Ajax proxy, multi-channel delivery, responsive layout, etc.

Build portlets using web development skills, libraries, and tools

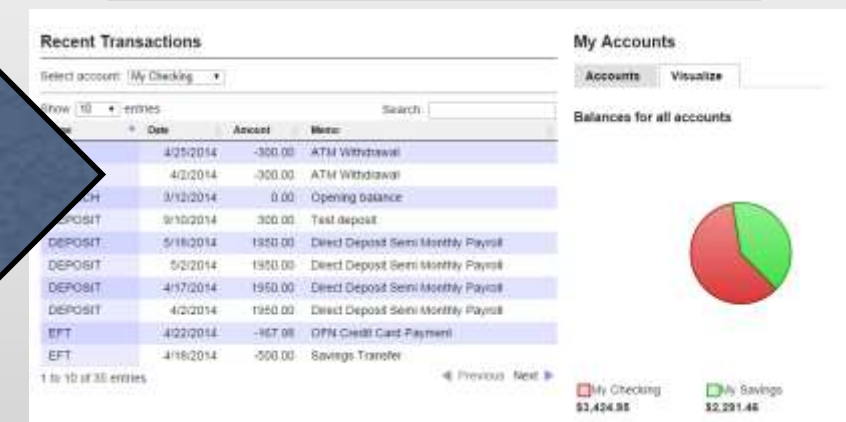
Use JavaScript libraries and frameworks of your choice



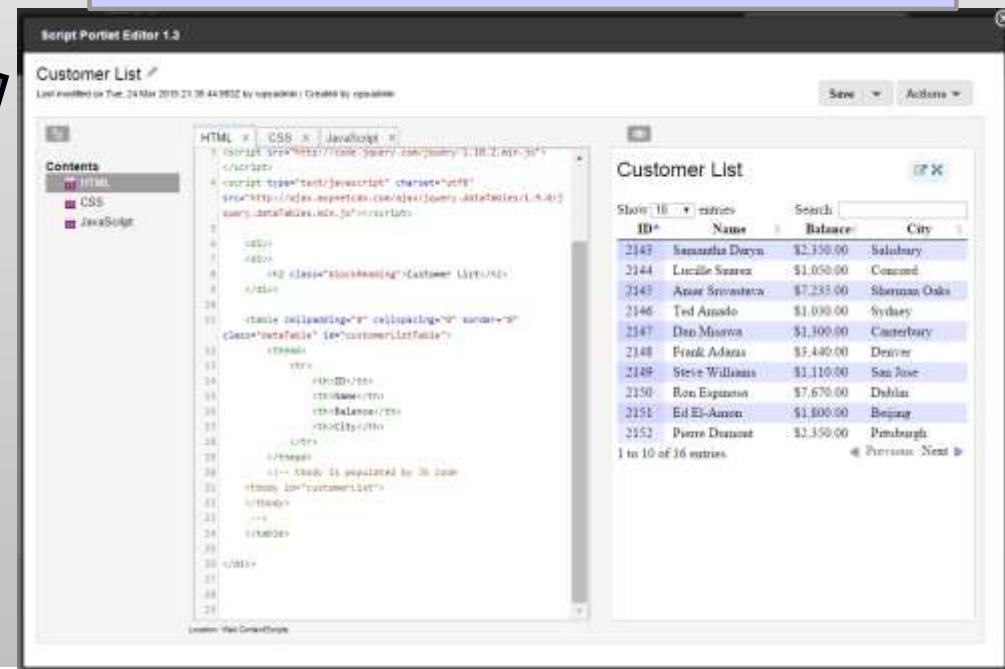
Applications are stored in Web Content Manager and available on site toolbar



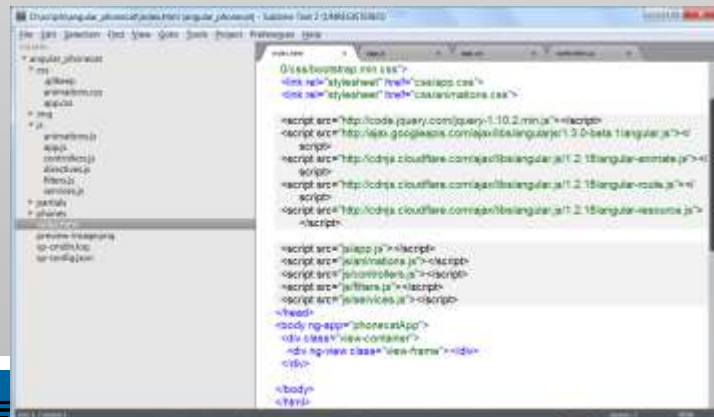
Rich portlet applications with support for Portal value-add features



Applications can be built and edited with browser-based editor



Optional: build/test applications with your preferred tools, then “push” or import to server



Approaches for building applications with Script Portlet

1. Build in the browser with the Script Portlet Editor
 - Type or copy/paste into the tabbed editor
 - Auto-complete and syntax highlighting are available
 - Applications are limited to single HTML, JavaScript, and CSS elements
2. Build on your workstation with your preferred tools, then use “sp push” or Import to move the application to the server
 - Applications can have any number of files including JS, images, HTML fragments, etc.
 - With “sp push” you can instantly update the entire application from a folder on your machine

In the next slides we'll show both approaches

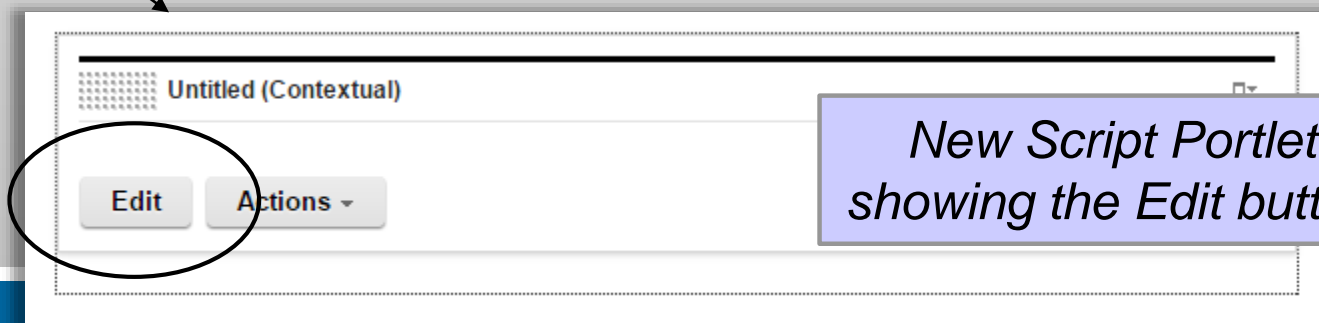
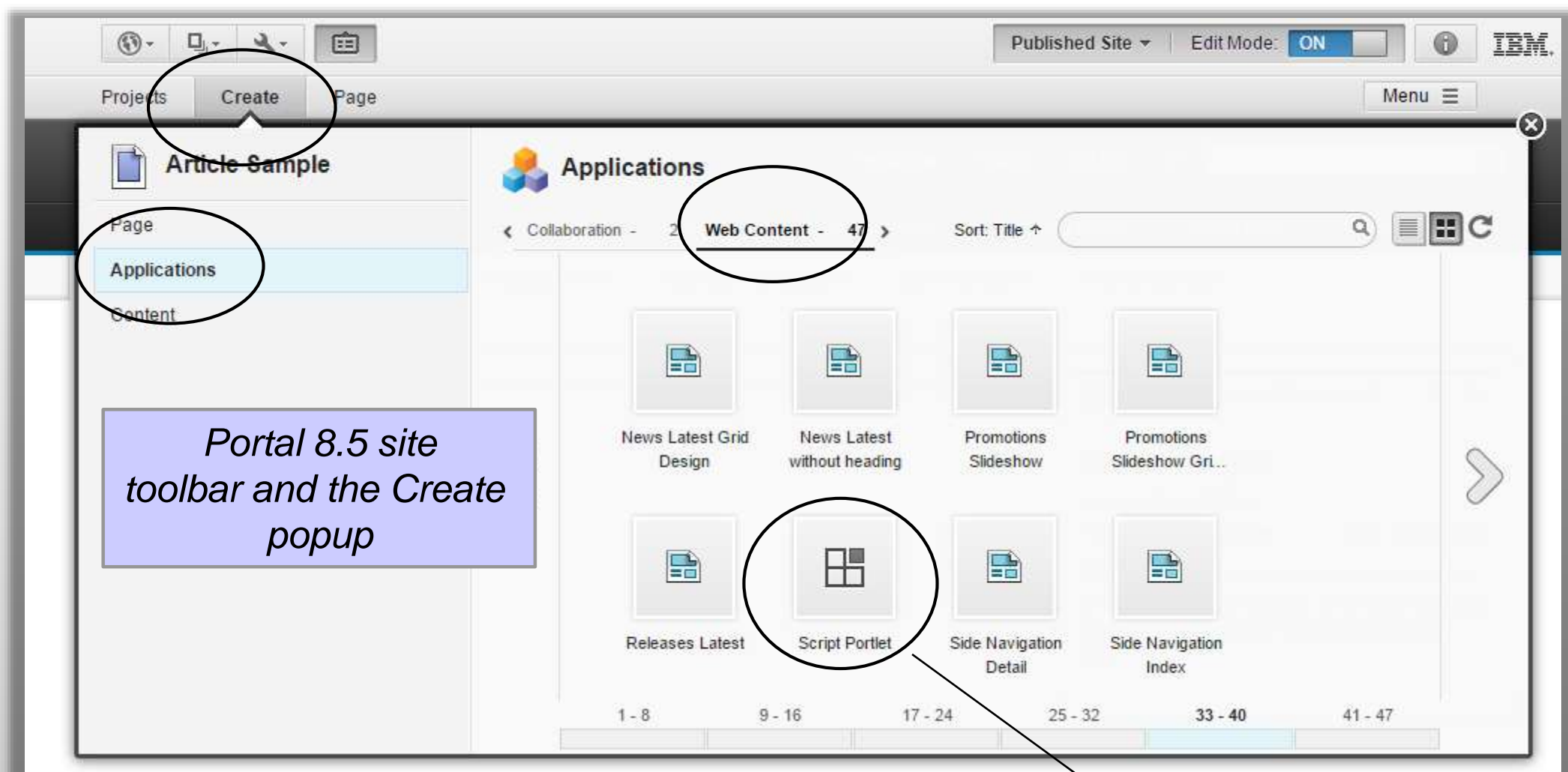
Building Applications with the Script Portlet Editor



Building applications in the browser using the Script Portlet Editor

- Create a new empty Script Portlet on a page
- Open the Script Portlet Editor and enter your HTML, JavaScript, and CSS
 - You can copy/paste from samples on the web, from JSFiddle, etc., or type your code
 - The HTML editor tab has any external references and all of your application UI
 - The JavaScript tab has your application JS code
 - The CSS tab holds your style definitions
- Click Save to preview the application
- Close the editor window to see the application on the portal page

Creating a new empty Script Portlet



The Script Portlet Editor with tree view, tabbed editors, and preview



The running portlet with the page in view mode

IBM® wpsadmin Actions Log Out ?

Welcome Demos Whats New Demos Samples Testing Published Samples Banking Home WEF Demos

Script Demo Script Demo 2 Toolbar Demo To Do Sample Customer List Samples List Banking Chart Banking Transactions

Demos > Script Demo

Customer List

Show 10 entries Search:

ID	Name	Balance	City
2143	Samantha Daryn	\$2,350.00	Salisbury
2144	Lucille Suarez	\$1,050.00	Concord
2145	Amar Srivastava	\$7,235.00	Sherman Oaks
2146	Ted Amado	\$1,030.00	Sydney
2147	Dan Misawa	\$1,300.00	Canterbury
2148	Frank Adams	\$3,440.00	Denver
2149	Steve Williams	\$1,110.00	San Jose
2150	Ron Espinosa	\$7,670.00	Dublin
2151	Ed El-Amon	\$1,800.00	Beijing
2152	Pierre Dumont	\$2,350.00	Pittsburgh

1 to 10 of 16 entries Previous Next

Article Sample

wpsadmin - Aug 27, 2014

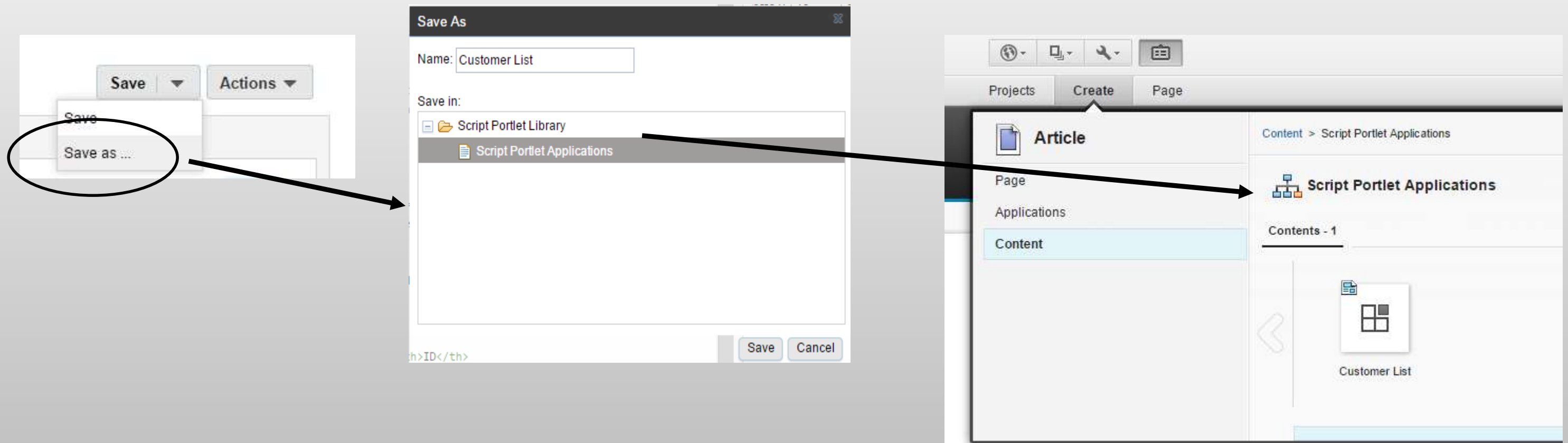
Tags: No tags + ☆☆☆☆☆

This is a place holder article that was copied into this page. To edit this article, ensure that the page is in edit mode and click the **Edit current item** icon.

Using “Save As” from editor to store in a library and add to toolbar

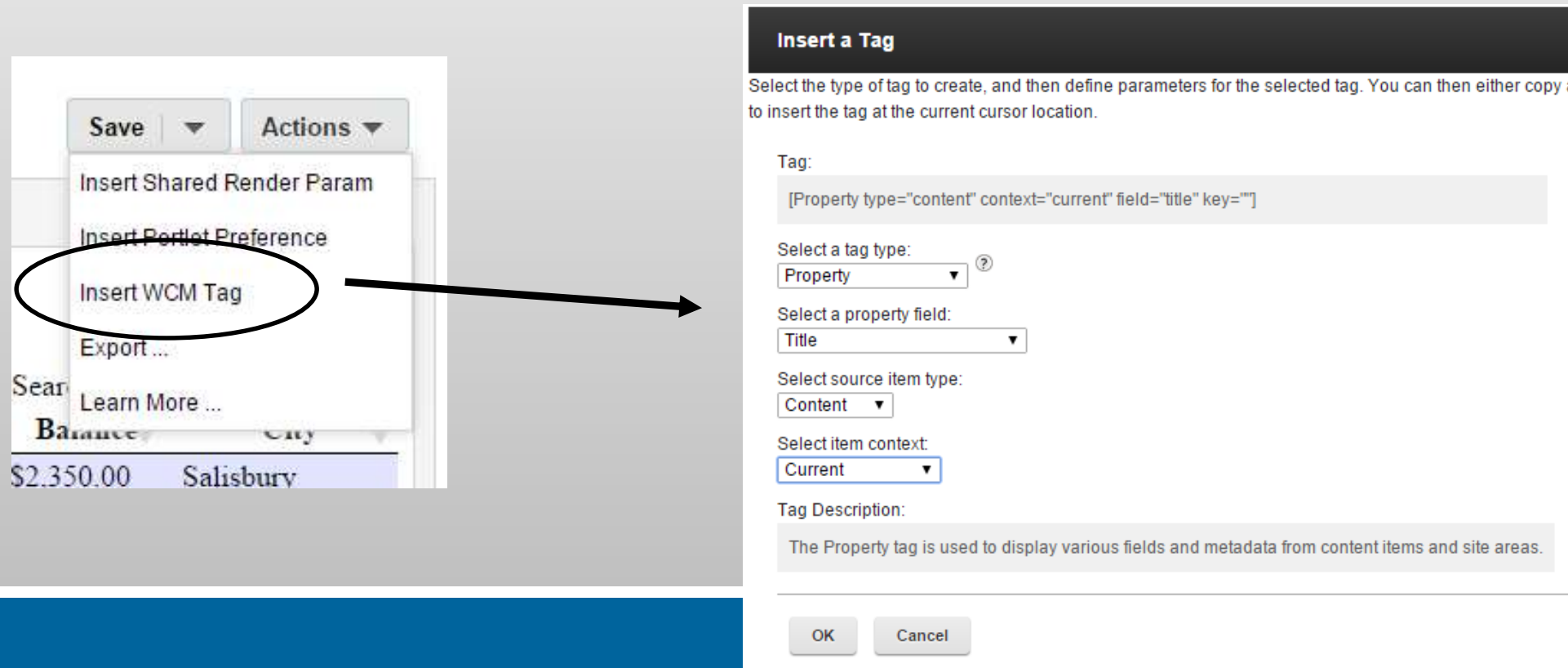
New
for 1.3

- The “Save As” command lets you save an application into a WCM library
- Applications in the library are available in the “Content” tab of the site toolbar for use by page authors



Inserting WCM tags from the Script Portlet editor

- Script Portlets can optionally use WCM tags to render dynamic elements and access portal and server-side features
 - Shared render parameters, device class, user attributes, and much more
- The “Insert WCM Tag” option brings up the standard WCM dialog and makes it easy to select a tag with parameters



Building Applications with External Editors and Tools

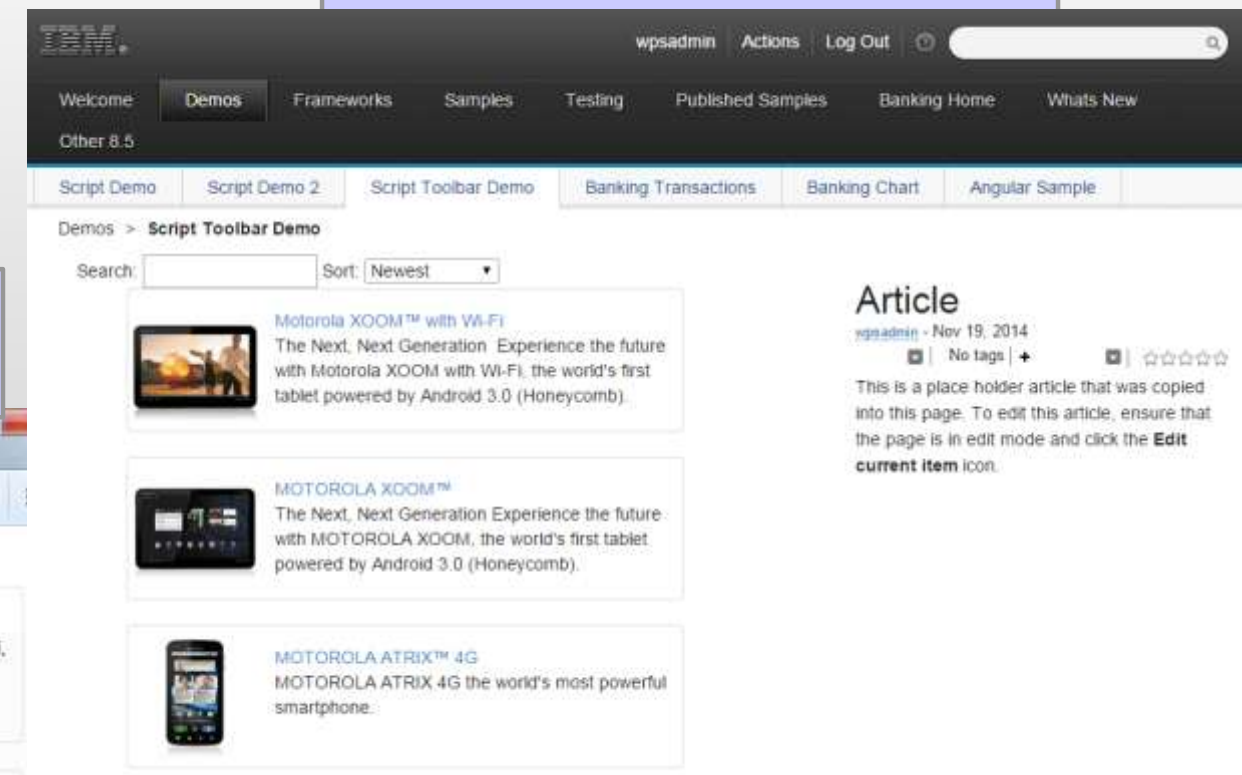
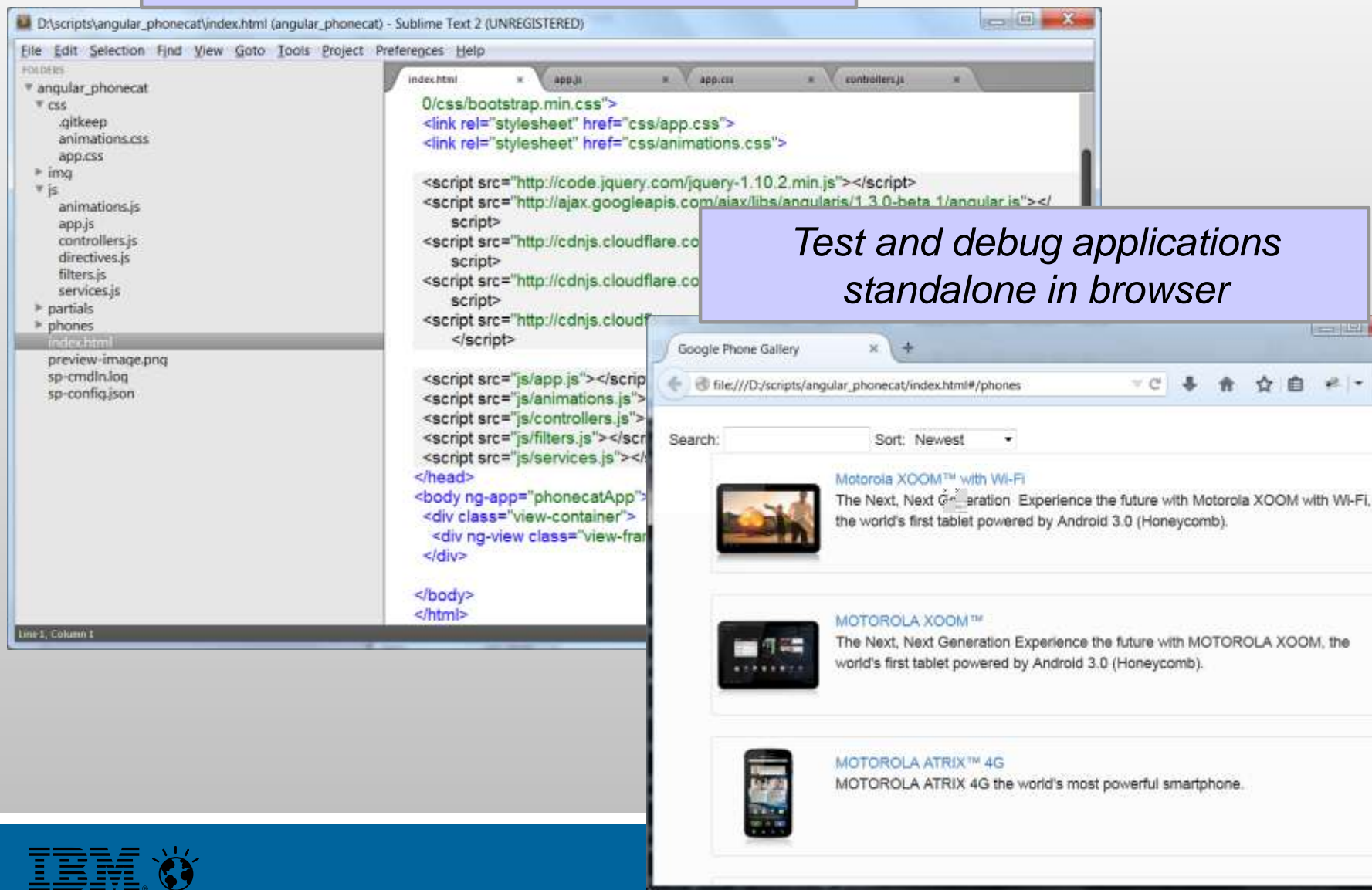


Use your favorite web development tools to build applications, then bring the applications into Script Portlet

Build simple or complex applications with your favorite editors and other tools such as Node.js-based tools

Push or Import the entire application folder into Script Portlet to run as portlet

Test and debug applications standalone in browser



Bringing web applications from a client machine into Script Portlet

Each web application should have a main HTML file (typically index.html) with other files such as JS, CSS, images, HTML fragments, JSON, etc.

You have two ways to bring these applications into Script Portlet:

1. **sp push:** Use this command from the client machine to **create** or update a script portlet
2. **Import:** Create a .zip for your application folder, then add an empty script portlet to a page and use the Import command

Create using sp push is new for 1.3

Both push and import can be used with local or remote/cloud servers

Using “sp push” to create or update a script portlet

New
for 1.3

Application folder with a main *index.html* and other *JS*, *CSS*, *HTML*, etc.

1. Invoke ***sp push -wcmContentName “<name>”*** from the application folder

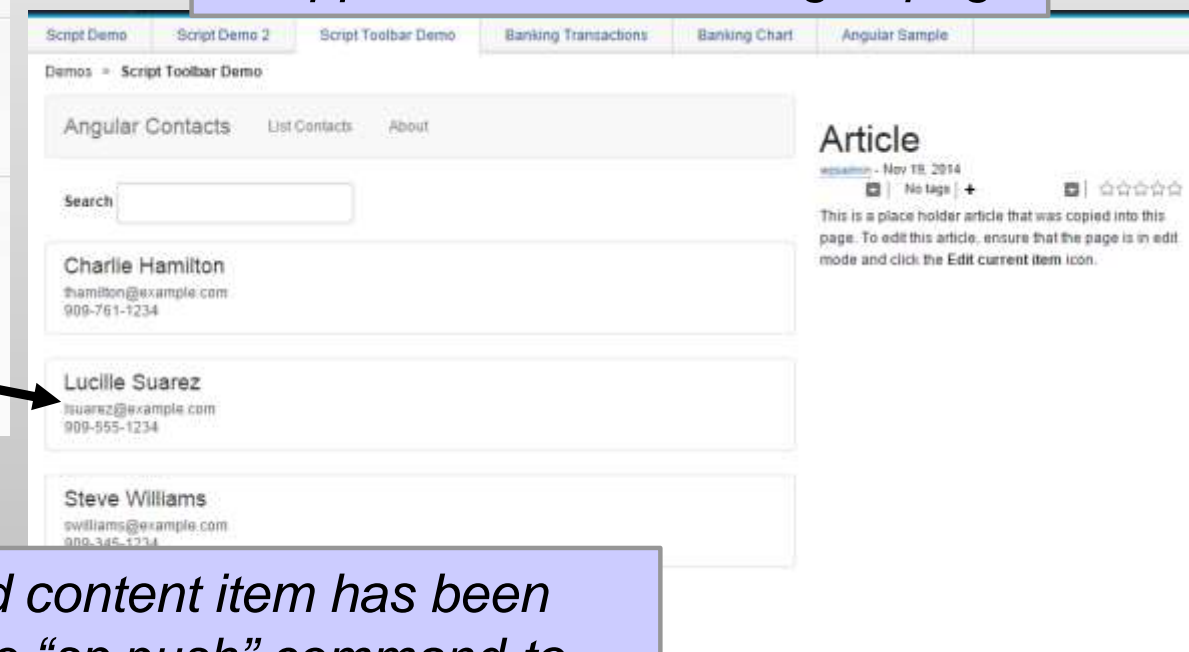
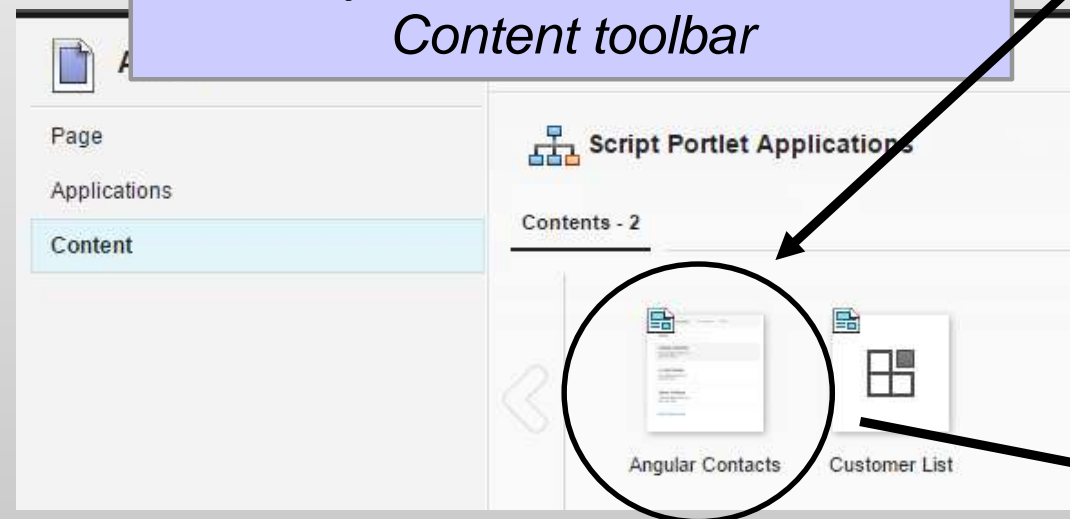
```
D:\published_scripts\angular_contacts>sp push -wcmContentName "Angular Contacts"
```

2. Application is added to a *WCM* library and is available on the *Content* toolbar

3. Application after adding to page

4. Once the specified content item has been created, use the same “*sp push*” command to update the existing content item

- ▼ angular_contacts
 - ▼ css
 - app.css
 - ▼ data
 - contacts.json
 - ▼ js
 - app.js
 - controllers.js
 - services.js
 - ▼ partials
 - aboutView.html
 - detailsView.html
 - listView.html
 - updateView.html
 - index.html




Multi-file application in Script Portlet editor after push or import



All files are shown in tree and can be opened for editing.


You can't create or delete files from this editor – instead, any file or folder changes on the client machine will be reflected when you push/import again.

Script Portlet Editor 1.3

Angular Contacts Sample 

Last modified on Fri, 20 Mar 2015 19:15:47.341Z by wpsadmin | Created by wpsadmin




Save  Actions 

 **Contents**


- HTML
 - preview-image.png
 - js
 - app.js
 - controllers.js
- services.js
- css
 - app.css
- partials
 - aboutView.html
 - detailsView.html
 - listView.html
 - updateView.html
- data
 - contacts.json

HTML x app.js x aboutView.html x contacts.json x

```
13 element="js/services.js" element="js/controllers.js"]
14 <script src="[Plugin:ScriptPortletElementURL
15 element="js/app.js"]"></script>
16 <script src="[Plugin:ScriptPortletElementURL
17 element="js/services.js"]"></script>
18 <script src="[Plugin:ScriptPortletElementURL
19 element="js/controllers.js"]"></script>
20 [ /Plugin:ScriptPortletURLCombiner] </div>
21
22 <div data-script-portlet-original-tag="body">
23 <div ng-app="contactsApp">
24
25 <nav class="navbar navbar-default">
26 <div class="container-fluid">
27 <ul class="nav navbar-nav">
28 <li role="presentation"><div class="navbar-
29 brand">Angular Contacts</div></li>
30 <li role="presentation"><a title="" target=""
31 href="#/list">List Contacts</a></li>
32 <li role="presentation"><a title="" target=""
33 href="#/about">About</a></li>
34 </ul>
35 </div>
36 </nav>
37
38 <div ng-view="">
39 </div>
40 </div>
41
```

 Angular Contacts List Contacts  

About

Search 

Charlie Hamilton
thamilton@example.com
909-761-1234

Lucille Suarez
lsuarez@example.com
909-555-1234

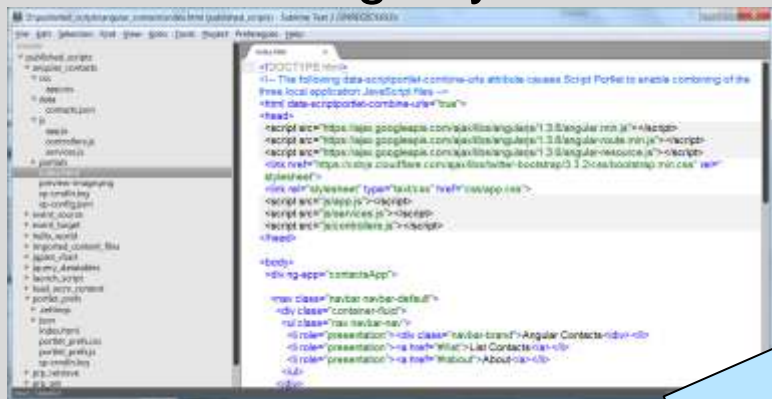
Steve Williams
swilliams@example.com
909-345-1234

[Reset Default Data](#)

Using “sp push” for a rapid edit/test cycle

Developer workstation

1. Edit files using any editor



2. Run sp push command

```
sp push -wcmContentName "Angular Contacts"
```

3. Refresh browser

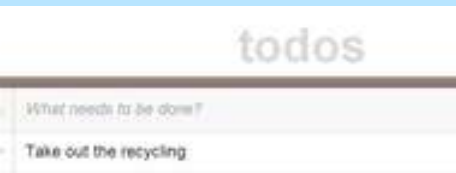


Portal server (local or remote)

WCM

WCM Content Item, with elements for application files (HTML/JS/CSS/images/etc.)

Portal page with Script Portlet

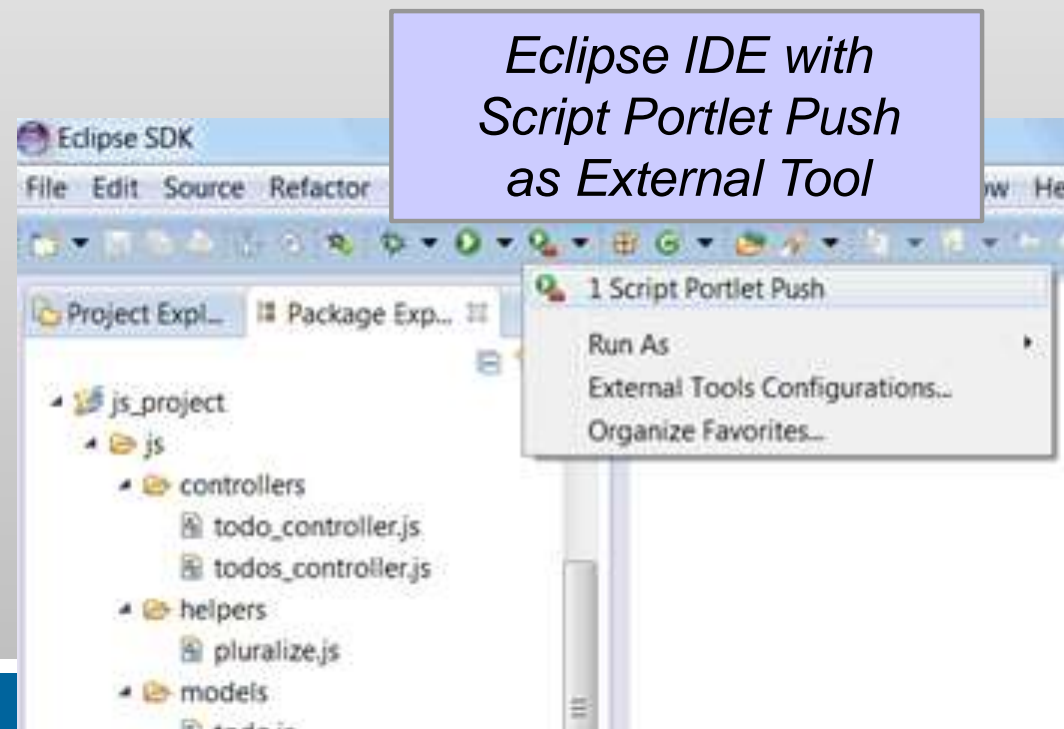


All files in folder are uploaded into WCM

Reloading the page renders all the latest code

Integrating command line support with editors and other developer tools

- Editors and developer tools typically provide a way to call external programs
- You can invoke “sp push” from a command or icon, or it may be automatically triggered by an operation such as saving a file
- For Node JS environments, “grunt” or “gulp” can watch for file changes and push an updated version whenever there’s a change



Sublime editor – key mapping for “sp push”

```
[
  {
    "keys": ["f1"],
    "command": "exec",
    "args": {
      "cmd": ["sp.bat", "push", "-contentRoot",
              "D:/test_scripts/angular_sample"]
    }
  }
]
```

Using the Import command with .zip files

You can also bring applications into Script Portlet using Import:

- Create a .zip file with all the contents of your application folder
- Create a new empty Script Portlet and select Import from the Actions menu
- The import processing and file handling is the same as for push

Note that this import .zip approach is often used as an easy one-time method for bringing applications from some other system, but it's not recommended for iterative edit/test – use “sp push” for that

Complex Applications, JavaScript Frameworks, and the “Single Page Application” Model

About complex applications, JavaScript frameworks, and the “single page application” model

- Complex web applications will often use JavaScript frameworks to structure the application and to provide valuable common functionality
 - Angular, Backbone, ExtJS, Ember, etc.
 - Frameworks may include features such as data binding, views, controllers, HTML templating, storage...
 - These frameworks can work very well with Script Portlet
- The “single page application” (SPA) model is often used
 - A main HTML page provides the shell where all interaction takes place, with no full page reloads
 - JavaScript handles all dynamic interaction and accessing data from server
 - Multiple views may be swapped in/out within the main HTML shell
- ***With Script Portlet, a “single page application” runs as one portlet on a portal page***
 - ***Don't be confused by the terminology – the portal site still has many pages***

Example of a single page application using the Angular framework, running standalone outside Portal



- angular_contacts
 - css
 - app.css
 - data
 - contacts.json
 - js
 - app.js
 - controllers.js
 - services.js
 - partials
 - aboutView.html
 - detailsView.html
 - listView.html
 - updateView.html
 - index.html

```
<html data-scriptportlet-combine-urls="true">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.6/angular.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.6/angular-route.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.6/angular-animate.js"></script>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet" type="text/css" href="css/app.css">
  <script src="js/app.js"></script>
  <script src="js/services.js"></script>
  <script src="js/controllers.js"></script>
</head>

<body>
  <div ng-app="contactsApp">

    <nav class="navbar navbar-default">
      <div class="container-fluid">
        <ul class="nav navbar-nav">
          <li role="presentation"><div class="navbar-brand">Angular Contacts</div></li>
          <li role="presentation"><a href="#/list">List Contacts</a></li>
          <li role="presentation"><a href="#/about">About</a></li>
        </ul>
      </div>
    </nav>

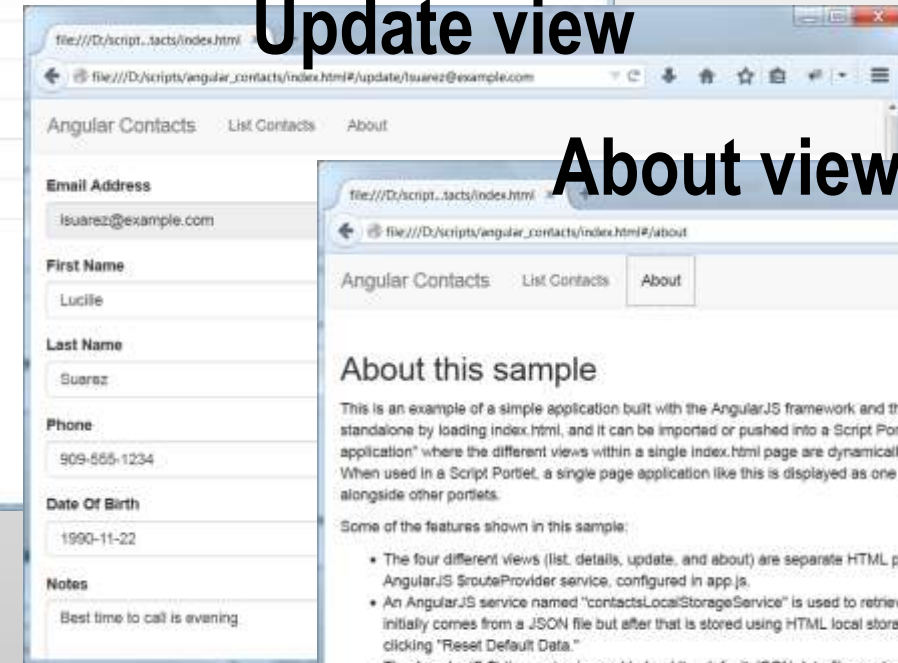
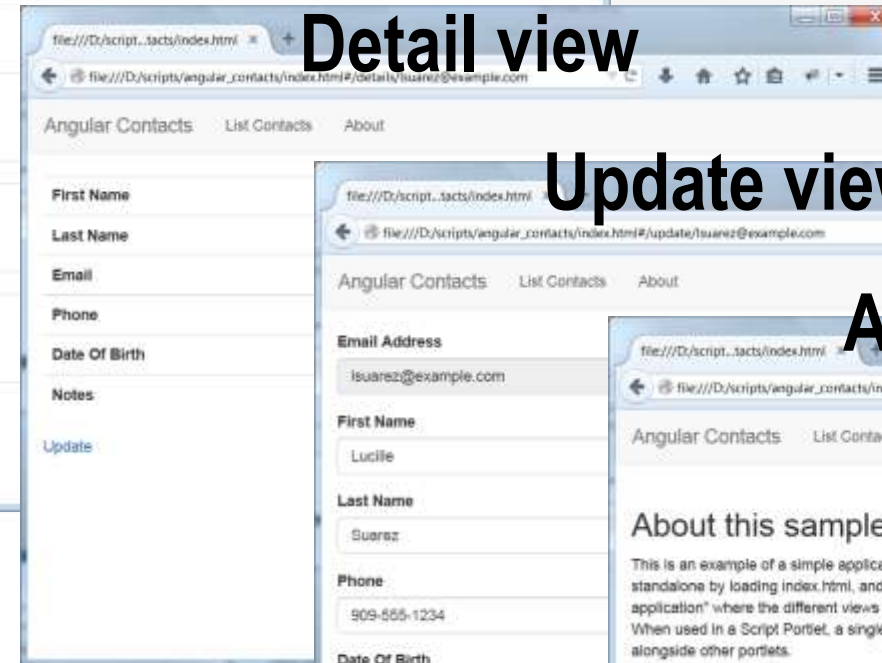
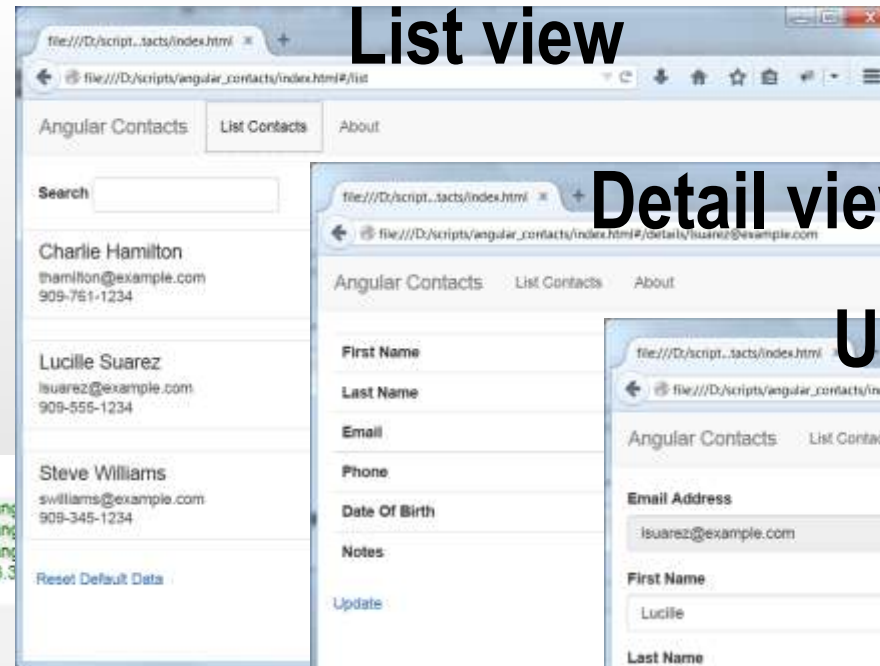
    <div ng-view="">
    </div>
  </div>
</body>
</html>
```

List view

Detail view

Update view

About view



The same example Angular application running as a script portlet, alongside other portlets or content

The “single page application” runs as one portlet on a portal page, along with other portlets or content

List view

Detail view

Update view

About view

*All the dynamic views
for the SPA will be
displayed within this
portlet area*

Handling of web application code with push or import

For both push and import, the handling of application files is the same:

- The main HTML file (typically index.html) must be at the top application folder – this will be rendered as a portlet
- You can have any number of other application files within the same folder or in sub-folders
 - JS, HTML fragments, images, JSON, media, etc.
- All the application files are stored as elements in one WCM content item
- Relative links to local files are converted to WCM tags for rendering
- Script Portlet Editor can be used to view and update any application files
 - ...but not to create or delete files

Tips for bringing web applications into Script Portlet

- Many single page web applications can be imported/pushed to Script Portlet without changing any of the code
- *This interoperability is an important goal of the Script Portlet tool*
- There are some key practices and techniques for this interoperability:
 - JS code shouldn't update the entire BODY tag, but instead should work within a DIV (most frameworks can support this)
 - CSS styling shouldn't be set for the body tag
 - JS libraries such as jQuery should be loaded from Portal theme or, for demo/POC, from a remote CDN link
 - New features in 1.3 can be used – see following slide
- Some explicit Portal dependencies such as WCM tags or the Script Portlet “spHelper” object won't work when running outside of Portal
- See articles on Digital Experience Developer site for details and best practices

New features available for use with push and import

- Automatic namespace support
 - Use the `__SPNS__` pattern in any HTML or JS to generate a Script Portlet instance ID when running
- Combining of application JS files for performance
 - Requires Portal 8.5 CF03 or later
 - Use the `data-scriptportlet-combine-urls="true"` attribute on your application's HTML element to enable this aggregation for a specific application, or you can enable it by default with a configuration property
- Automatic removal of external links for libraries that are in theme modules
 - Use the `data-scriptportlet-theme-capability="name"` attribute to mark a link for removal during push/import, for cases where you know it will be provided by your Portal theme
- JS helper support for dynamically-loaded application files
 - In certain scenarios where application relative links are generated in JS code, you may need a way to map the original local URLs to WCM URLs
 - Use the `data-scriptportlet-generate-url-map="true"` attribute to enable support for this mapping in the `<namespace>spHelper` object
- Support for media and other binary file types
 - Several media types are enabled by default, and you can set configuration properties to allow additional file types
- *See product documentation and articles on developer site for details on these features*

Making Applications Available on the Site Toolbar



Making applications available on the Content tab of site toolbar (Portal 8.5 only)

New
for 1.3

- Any applications that are stored under Script Portlet Library / Script Portlet Applications will be available on the site toolbar in the “Content” tab
- You can store applications in this area with **sp push** or with the Script Portlet editor “Save As” command
- “Script Portlet Applications” is the default site area for storing Script Portlets in a library, but you can use child site areas or other locations of your choosing – see the documentation for details
- You can provide a custom icon for each application by including a .png/.jpg file named “preview-image” at the root of your application folder

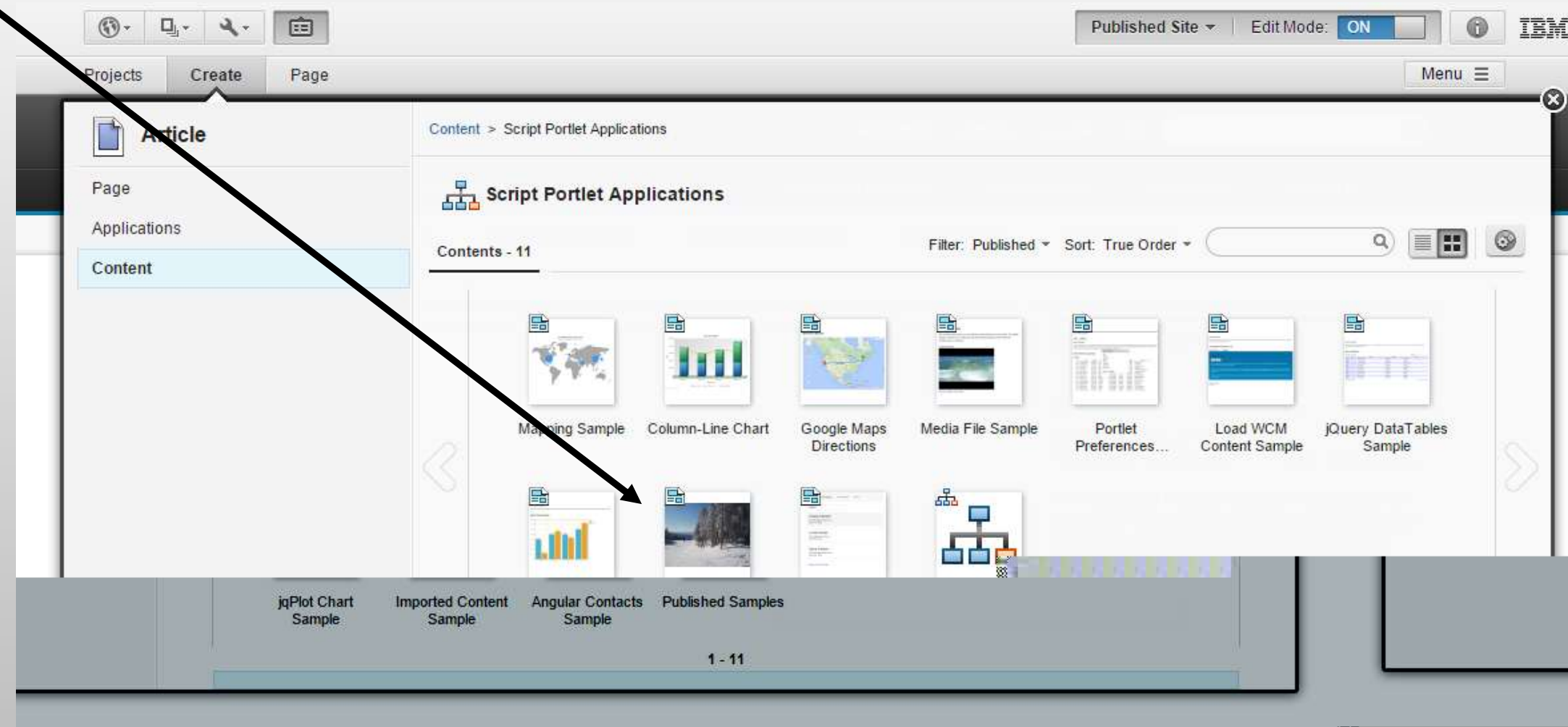
Example applications on toolbar, with custom icon images

New
for 1.3

*preview-image.png for
display on the site
toolbar*

- ▼ imported_content_files
 - ▼ html
 - testFragment.html
 - ▼ images
 - skiing.jpg
 - ▼ json
 - customerData.json
 - app.css
 - app.js
 - index.html
 - preview-image.png
 - sp-config.json

*sp-config.json file has
wcmContentName set
("Imported Content
Sample")*



About links vs copies for applications added from toolbar

- When you drag or add an application from the toolbar, it will result in a link reference to the shared content item
- Any updates to the shared item will be reflected wherever the application is used
 - For iterative editing with an external editor, you can use “sp push” to update the shared item and instantly see your changes on a page
- This “link” behavior requires CF05 or later – prior versions of 8.5 will generate a copy
- To make a new copy from the Script Portlet editor you can use Save As

Using Command Line Features and Working with Source Code Systems

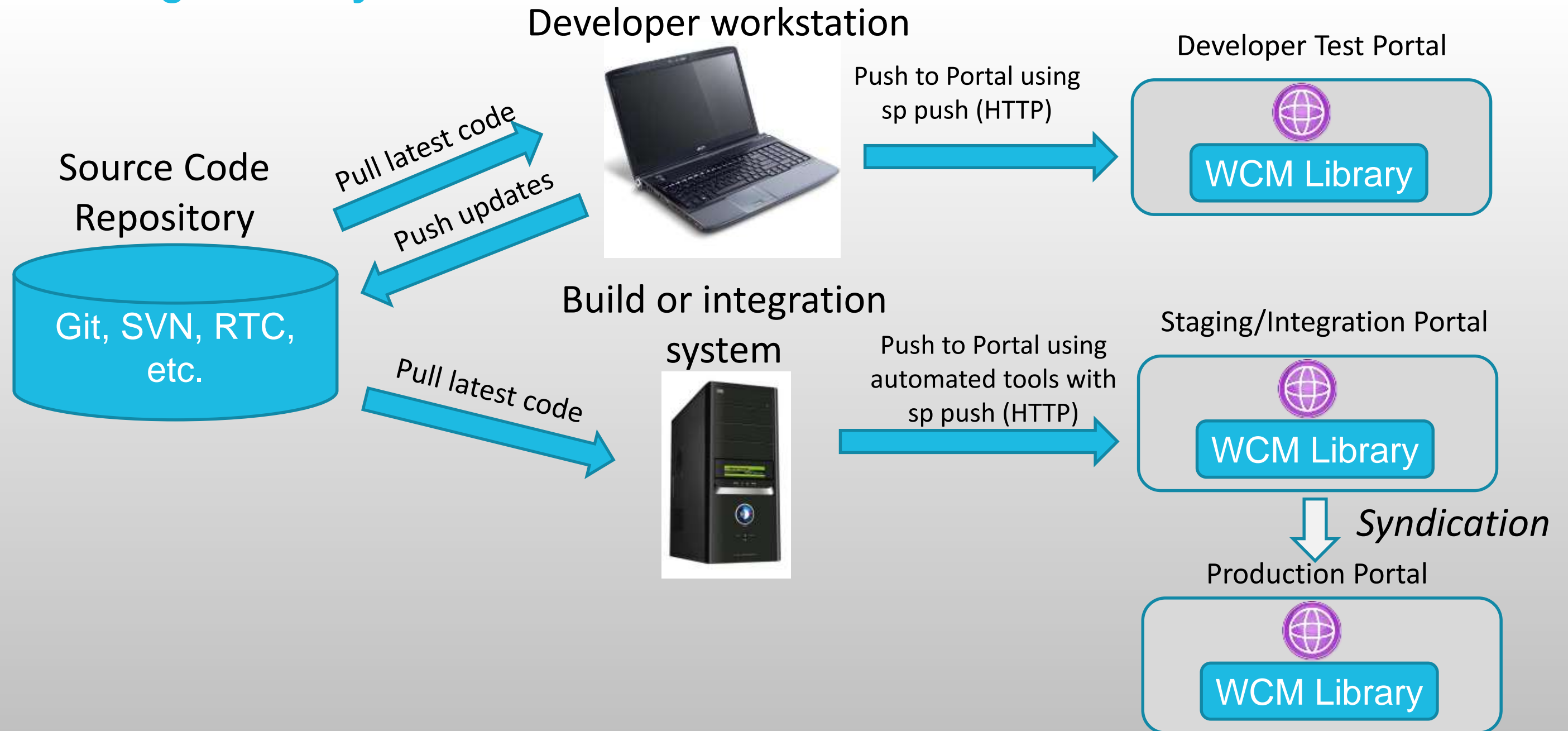


Working with source code systems

- Using “sp push” you can automatically populate a local or remote WCM library with a set of applications on a file system
- You can keep all your Script Portlet application code in a source code management system and automatically push the latest code into WCM
- For moving applications between portal servers (such as from staging to production) you can also use WCM’s syndication feature
- See the following slide for a picture

Synchronizing Script Portlet application code with source code management systems

New
for 1.3



Using sp push and the sp-config.json configuration file

- An sp-config.json file is used to set properties for sp push
- Any of the arguments for the sp push command (such as wcmContentName) can also be specified in sp-config.json
 - Type “sp help” to see details on the arguments
- There is a shared sp-config.json file in the command line client folder, and you can also have an sp-config.json file in any application folder
 - Common values such as the properties for your test server are set in the shared file
 - Values from the application-specific sp-config.json file will override the shared values
 - For example, by setting wcmContentName in the application-specific sp-config.json file you can omit it from the command line and ensure consistent naming
- See the documentation or run “sp help” for information on other properties such as wcmSiteArea, wcmContentTitle, virtualPortalID, and others

Quick Steps for Getting Started and Downloadable Samples



Five quick steps for getting started with Script Portlet 1.3

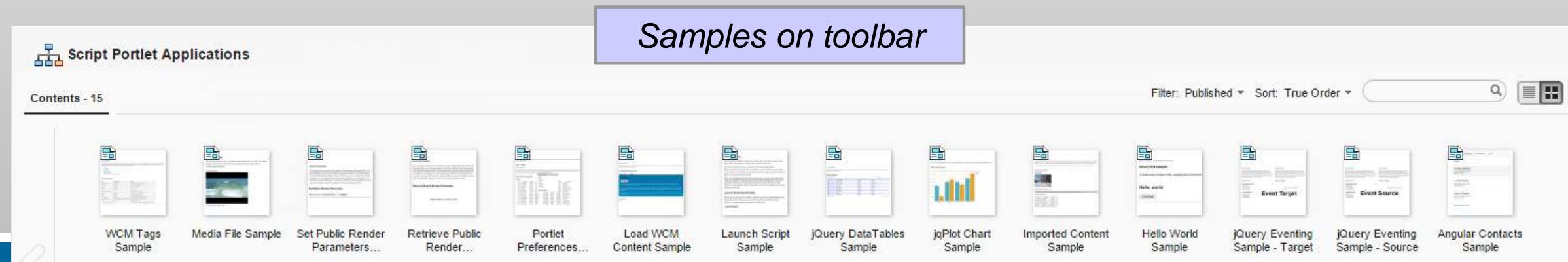
1. Use Portal 8.5 or 8.0.0.1 (local or remote/cloud)
 - 8.5 with CF05 or later is recommended because of the toolbar support
2. Install the Script Portlet PAA and confirm installation by adding a Script Portlet to a page and using the Script Portlet editor
 - Download: [IBM Script Portlet on Portal Catalog](#)
 - Install documentation: [Installing the IBM Script Portlet](#)
3. Extract and configure the command line client support (sp_cmdln.zip) on your workstation
 - Add the extracted folder to your system path and type “sp” to confirm Java version
 - Set the properties for your server in sp-config.json
4. Download and extract the samples package, and use the batch file provided with the samples to push them into WCM and make them available on the toolbar
 - Samples package: [Script Portlet Samples for IBM WebSphere Portal](#)
5. Try out some of the samples by adding them to pages, and explore the source in the Script Portlet editor or in the extracted samples folder

Downloadable samples for Script Portlet

Downloadable samples are available that illustrate a variety of useful techniques

Run one command script to push them all to server and make them available on toolbar

- AngularJS Contacts (with Bootstrap styling) ★new
- Media File (.mp4 video) ★new
- Hello, World
- jQuery Eventing (two cooperating portlets)
- jqPlot Chart
- jQuery DataTables
- Load WCM Content
- Launch Script (launching a Script Portlet in a window)
- Public Render Parameters (two cooperating portlets)
- WCM tag samples (access to information such as user name, locale, device class)
- Imported Content Files (image, HTML fragment, JSON)
- Portlet Preferences (for customizable applications)



Downloadable sample screenshots

Samantha Daryn

Status: Standard
Updated: 07/01/2012

\$2,350.00

Lucille Suarez

Status: Preferred
Updated: 11/21/2

\$1,050.0

Amar Sriv

Status: Inner Circ
Updated: 08/12/2

\$7,235.0

Charlie Hamilton

thamilton@example.com
909-761-1234

Lucille Suarez

lsuarez@example.com
909-555-1234

Steve Williams

swilliams@example.com
909-345-1234

Angular Contacts

List Contacts

About

Search

Reset Default Data

Customer Details

Lucille Suarez

2144

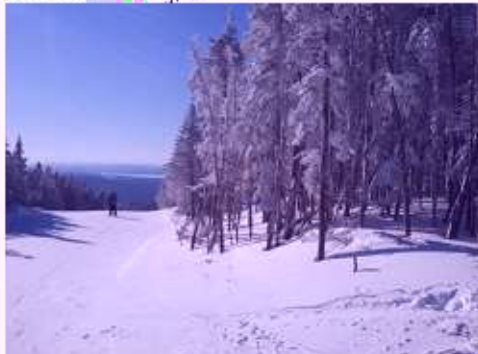
303-555-2435
123 Main St
Concord

Updated: 11/21/2012

Print Document

Imported Content Files

Imported



HTML Fragment Loaded by JS

This text comes from a separate HTML fragment file, dynamically loaded.

JSON Data Loaded by JS

Name	Address
Samantha Daryn	2001 West Rd.
Lucille Suarez	123 Main St

Select columns

☒ ID

☒ Name

☐ Balance

☒ City

☐ Phone

☒ Status

☐ Updated

☐ Address

Save

Cancel

jqPlot Chart Sample

Month	Goal	Actual
January	\$4,000	\$1,233
February	\$5,000	\$5,613
March	\$5,000	\$4,302
April	\$6,000	\$7,230

Set Public Render Parameter

Select customer:

Frank Adams

Submit

About this sample

This sample shows how to set and retrieve a Public Render Parameter using the WCM Rendering Portlet. The top portlet shows a list of customers and sets the PRP value. The bottom portlet then retrieves the PRP value.

Retrieve Public Render Parameter

Frank Adams

\$3,440.00

601-555-8888
133 Rock Boulevard
Denver

1 to 10 of 16 entries

Previous

Next

Customizable Applications and Accessing Data and Services



Enabling customization of applications using portlet preferences

- You can use portlet preferences to provide application customization at the portlet instance level
 - Applications can be configured from the browser by business users, page authors, or administrators
 - Preference data is stored as JSON
- The JavaScript object `<namespace>spHelper` has functions to store and retrieve preference data
- See the published samples for an example

Examples of customization using preferences API

- A “Stock Tracking” application can be configured for which stocks to show

Stock Tracking Application

Settings

International Bus

IBM

185.46

-0.47

-0.25%

Google Inc.

GOOG

541.19

-3.30

-0.61%

Apple Inc.

AAPL

100.81

+0.08

+0.08%

Oracle Corp.

ORCL

38.51

+0.41

+1.06%

Microsoft Corpora

MSFT

27.12

-0.15

-0.55%

Enter stocks

Enter up to 10 symbols for stocks you want to track.

Symbol

IBM

GOOG

MSFT

AAPL

ORCL

FB

Save

Cancel

Restore to initial

A customer list can be configured for which columns to display

Customer List

Settings

ID	Name	City	Status
2143	Samantha Daryn	Salisbury	Standard
2144	Lucille Suarez	Concord	Preferred
2145	Amar Srivastava	Sherman Oaks	Inner Circle
2146	Ted Amado	Sydney	Standard
2147	Dan Misawa		
2148	Frank Adams		
2149	Steve Williams		
2150	Ron Espinosa		
2151	Ed El-Amon		
2152	Pierre Dumont		
2153	Gardner Raynes		
2154	Dennis Michaels		
2155	Suzanne Miles		
2156	Betty Zechman		
2157	Jasmine Haj		
2158	Charlie Hamilton		

Select columns

☒ ID

☒ Name

☐ Balance

☒ City

☐ Phone

☒ Status

☐ Updated

☐ Address

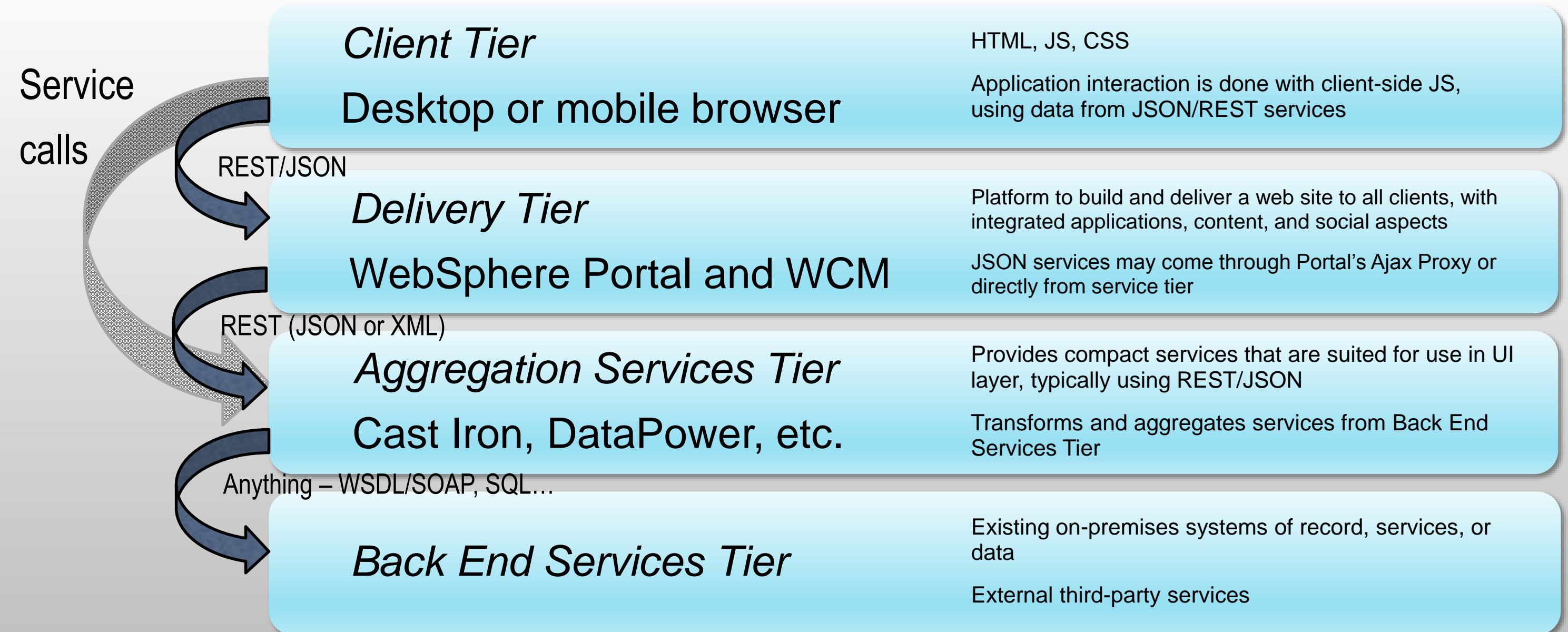
Save

Cancel

Accessing data and services

- Data access for script-based applications is done using REST services with JSON format data
- Any number of tools, frameworks, and services can provide the REST/JSON services:
 - Cloud-based services, including IBM SmartCloud and Bluemix services
 - IBM Cast Iron platform or DataPower appliance
 - IBM Web Experience Factory data providers
 - WAS connectors (on Portal server or external)
- External services can be accessed through Portal's Ajax proxy
- The current trend for application architecture is to have a four-tier architecture (see next slide)

Four tier architecture and data services



Data service example: accessing bank account data

Banking Transactions Table

Select account: My Checking ▾

Show 10 ▾ entries

Search:

Type	Date	Amount	Memo
TRANSFER	7/11/2014	-572.87	My Checking
DEPOSIT	5/18/2014	1950.00	Direct Deposit Semi Monthly
PAYMENT	5/16/2014	-17.96	Shell
PAYMENT	5/15/2014	-45.73	Corner Grocery Store
PAYMENT	5/11/2014	-16.59	BP
PAYMENT	5/8/2014	-42.19	Corner Grocery Store
PAYMENT			
PAYMENT			
DEPOSIT			
PAYMENT	5/1/2014	-43.09	Corner Grocery Store

Recent bank transactions for selected account, using data from REST/JSON service, displayed with jQuery Datatables

1 to 10 of 34 entries

◀ Previous Next ▶

File ▾

Actions ▾

JavaScript

```
33 }
34 }
35
36 function displayTransactions(id, accountNumber) {
37     // Calls REST URL, with accountNumber as input
38     $.getJSON( bankingRestUrls.getTransactionsURL,
39         {'account-oid': accountNumber},
40         function(data) {
41             displayDataTable(id, data.RowSet.Row);
42         }
43     );
44 }
```

JS code uses jQuery getJSON function to call REST service, using accountNumber as input

For more information

- Visit IBM Digital Experience Developer:
<http://developer.ibm.com/digexp>
- Links to download of Script Portlet, samples, documentation, video, and other resources

